Control system for Robotic Arms over Controller Area Network (CRACAN)

[1]Anu Jose, [2]Shinu M R, [3]Gana Rani Jose

[1,2,3]Department of Computer Science Engineering,
Amrita Vishwa Vidyapeetham, Bangalore
Email: [1]Anujoseph.me@gmail.com, [2]mr_shinu@blr.amrita.edu, [3]rani.gana@gmail.com

Abstract - Now a day's robots have advanced from single processor to a networked multiprocessor system. However, sequential and synchronous control methods that traditionally have been adapted to robots and automation systems are not functioning well for distributed robots and control systems. In this work a distributed robot control system based upon controller area network (CAN), of which components are implemented with mostly aperiodic and event triggered approaches to exchange messages is discussed.

Keywords: robot control system, arm robots, keil compiler, Controller area network, event triggering, adaptive cruise control, ECU, Windshield Wiper control, PI controller.

I. INTRODUCTION

The most known network protocols for embedded systems are event-triggered (ETP) and time-triggered (TTP), which provide their own advantages of using them. Currently event-triggered protocol is used in vehicles, which means that in principle all activities are invoked by an event; most messages are transmitted periodically to the communication bus though. The event-triggered protocol is non-deterministic because the activities of a bus system are advanced by the sequence of events. In addition messages are exchanged in response to events spontaneously. In Event-triggered vehicular systems ECUs, sensors and actuators all are demand driven. The ECUs processes upon receiving sensor signals, in which a sensor with a changing value immediately sends a message to the ECU. The actuators behave upon signals from the ECUs. The time-triggered protocols are deterministic because the tasks are executed by the progression of time, and messages are exchanged periodically.

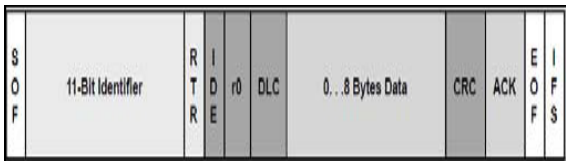
In time-triggered automotive systems, sensors are polled by ECUs regularly to provide any changes or signals, upon which ECUs execute the corresponding tasks. The actuators are working periodically even without signals from ECUs. The event-triggered system requires a dynamic scheduling strategy because the time of a task invocation cannot be predicted

In distributed robots environment, various components such as sensors, actuators, computers, and users are demanded to work in a good harmony and coordination in order to allow multiple robots to operate synchronously or asynchronously. Hence this enables the multiple robots over a communication network to operate seemingly on distributed control environments, where tasks are scheduled to perform trying to achieve utmost efficiency in time usage and resource share. When selecting a communication network for networked or distributed robots environments, there are some expected aspects to look into such as bus access type, bandwidth, control option, and number of components available over the bus, power level, and others.

II. CONTROLLER AREA NETWORK

The Controller Area Network (CAN) is a multi master serial bus that uses broadcast to transmit to all the CAN nodes. CAN protocol have several advantages over other communication protocols. For example, CAN protocol offer a very good price/performance ratio. It allows moving data with a fast transmission speed (up to 1 Mbit/s) and can be implemented in real-time systems. Moreover, the data is very reliable and error detection is sophisticated and robust. The CAN bus was developed as a multi-master, message broadcast system that specifies a maximum signalling rate of 1M bit per second (bps). Unlike a traditional network such as USB or Ethernet, CAN does not send large blocks of data point-to-point from node A to node B under the supervision of a central bus master. In a CAN network many short messages like temperature or RPM are broadcast to the entire network, which allows for data consistency in every node of the system.

A. Frame format of CAN 2.0.



The meaning of the bit fields of Figure are

- SOF—The single dominant start of frame (SOF) bit marks the start of a message, and is used to synchronize the nodes on a bus after being idle.
- Identifier—The Standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority.
- RTR—The single remote transmission request (RTR) bit is dominant when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way all data being used in a system is uniform.
- IDE—A dominant single identifier extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.
- r0—Reserved bit (for possible use by future standard amendment).
- DLC—The 4-bit data length code (DLC) contains the number of bytes of data being transmitted.
- Data—Up to 64 bits of application data may be transmitted.
- CRC—The 16-bit (15 bits plus delimiter) cyclic redundancy check (CRC) contains the checksum (number of bits transmitted) of the preceding application data for error detection.
- ACK—Every node receiving an accurate message overwrites this recessive bit in the original message with a dominate bit, indicating an error-free message has been sent. Each node acknowledges (ACK) the integrity of its data. ACK is 2 bits, one is the acknowledgement bit and the second is a delimiter.
- EOF—This end-of-frame (EOF) 7-bit field marks the end of a CAN frame (message) and disables bit–stuffing, indicating a stuffing error when dominant. When 5 bits of the same logic level occur in succession during normal operation, a bit of the opposite logic level is stuffed into the data.
- IFS—This 7-bit inter-frame space (IFS) contains the amount of time required by the controller to move a correctly received frame to its proper position in a message buffer area.

B. Bus arbitration

Bus access is event-driven and takes place randomly. If two nodes try to occupy the bus simultaneously, access is implemented with a non-destructive, bit-wise arbitration.

Non destructive means that the node winning arbitration just continues on with the message, without the message being destroyed or corrupted by another node. The allocation of priority to messages in the identifier is a feature of CAN that makes it particularly attractive for use within a real-time control environment. The lower the binary message identifier number, the higher its priority. An identifier consisting entirely of zeros is the highest priority message on a network since it holds the bus dominant the longest. Therefore, if two nodes begin to transmit simultaneously, the node that sends a zero (dominant) while the other nodes send a one (recessive) gets control of the CAN bus and goes on to complete its message. A dominant bit always overwrites a recessive bit on a CAN bus.

III. LITERATURE SURVEY

CAN is a protocol for short messages. Each transmission can carry 0 - 8 bytes of data. This makes it suitable for transmission of trigger signals and measurement values. It is a CSMA/AMP (Carrier Sense Multiple Access / Arbitration by Message Priority) type of protocol. Thus the protocol is message oriented and each message has a specific priority according to which it gains access to the bus in case of simultaneous transmission. An ongoing transmission is never interrupted. Any node that want to transmit message waits until the bus is free and then starts to send the identifier of its message bit by bit. A zero is dominant over a one and a node has lost the arbitration when it has written a one but reads a zero on the bus. As soon as a node has lost the arbitration it stops transmitting but continues reading the bus signals. When the bus is free again the CAN Controller automatically makes a new attempt to transmit its message. This procedure as well as error checking and retransmission of corrupted messages are done by the CAN Controller chips. The arbitration procedure requires that there are a limited number of identifiers in Extended CAN more than 500 million) and that a specific identifier is sent only by one node. The only

exception from this rule is when a message carries no data. As the amount of data that can be sent in one transmission is limited to eight bytes the maximum latency time of the highest priority message can be calculated. The maximum latency time of any message can be calculated if the nodes are restricted to the use of the same message identifier, once transmitted, until a specified time has elapsed. Every CAN Controller in a network will receive any message transmitted on the bus. Each node has to check whether a message is for him or not. Any CAN Controller on the market offers some filtering capacity to reduce the processor capacity needed for this activity, some more elaborate than others. CAN was designed for event driven systems but it is not difficult to use the protocol in time driven systems. Systems mixing both principles are also possible.[3].

The main advantage of event-triggered systems is their ability to fastly react to asynchronous external events which are not known in advance. Thus, they show a better real-time performance in comparison with time-triggered systems. In addition, event-triggered systems possess a higher flexibility and allow in many cases the adaptation to the actual demand without a redesign of the complete system. Within a time-triggered communication the permission to access the bus is controlled by predefined time windows (TDMA, time division multiple access). Therefore, time-triggered concepts potentially provide a higher dependability, since e.g. missing messages are immediately detected. Other important properties are the possibility to guard the bus against non authorized bus accesses (bus guardian) and to realize synchronously working busses in order to take care for redundancy. A very interesting property from the point of view of the automotive field concerns the so called compos ability. Since the time windows to access the bus are predefined, the behaviour along the timeline is decoupled from the actual bus load. Thus, it is possible to develop different subsystems independently, to exactly simulate the final time behaviour of the subsystems and subsequently to integrate the subsystems in to the complete system. In general, reality is neither black nor white but rather gray. Thus, it depends on the application whether a time-triggered or an event-triggered behaviour is more suitable.[2]

Computing infrastructures of mobile robots have grown in complexity in the last decades; they have evolved from single processor systems to networks of microcontrollers communicating through a shared bus. This has induced additional architectural constraints that do not fit well with the traditional polling-based sensors and actuators control. To address this issue, they have developed ASEBA, an event-based middleware that allows distributed control and efficient resources exploitation of multi-microcontrollers robots. ASEBA provides hardware modularity, better efficiency, and improved scalability by embedding a lightweight virtual machine in each microcontroller and providing an IDE to develop and debug the whole robot reactive control from a single place.[4].

Scheduling messages on a CAN bus is analogous to scheduling tasks by fixed priorities. Because CAN messages are non-pre-emptive, the existing worst-case response time analysis for fixed-priority pre-emptive scheduling (FPPS) has been updated to take account of tasks being non pre-emptive, i.e. resulting in worst-case response time analysis for fixed-priority non-pre-emptive scheduling (FPNS).

IV. SYSTEM DESIGN

A. 4.1 Block diagram

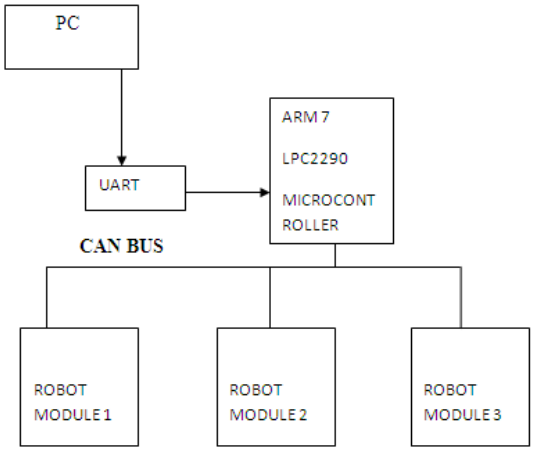


Fig 1: Block diagram of system

CRACAN is the design and development of a robot control system in which components are handled with event triggered approach over CAN (Controller Area Network). CAN bus is used to connect distributed hardware and software modules. CAN has been known to be very suitable any real time systems with its low cost and high reliability as a network. With CAN the functions of the distributed control systems perform well with more enhanced modularity and provide well-organized distributed controls.

In distributed control systems, conventional location based networks are not in favour, but rather message based networks such as CAN are in more favour. CAN has been known to be very suitable any real time systems with its low cost and high reliability as a network. Furthermore, the amount of wiring between components or modules is

drastically reduced by using a shared data bus, CAN, instead of hardwired point-to-point connections. As shown in Figure 1, CAN controllers and other components are connected over a shared CAN bus so as to avoid the need of having the components with point to point connections that accrue a large amount of wiring, more complex electric circuits and noise, which result in a less effective and reliable system.

With CAN, the functions of the distributed control systems perform well with more enhanced modularity and provide well-organized distributed controls. Under this setup the proposed distributed control system with CAN is able to perform and show the speedy synchronization The CRACAN system works with the following functions. End users can select the options on the main menu as a guide, where the menu options are shown: Hard-home all robots, Move to point A, Move to point B, Perform coordinated task, Perform sequential task, Open gripper, Close gripper, Check system status, Direct command, and Quit. More required functions are followed by such as:

- Hard-home all robots: This performs the Rhino hard home operation, where the Rhino controller positions the arm into an initial "home" position. This is a normally a pre-condition to beginning any robot tasks.
- Move to point A: This moves a user-named robot to position "A".
- Move to point B: As above, but using position "B".
- Perform coordinated task: This option simulates a coordinated job, representative of an assembly line. All robots perform simultaneously. The user supplies the number of repetitions of the "job" to be performed, which consists of moving to point A, where the gripper is closed, to point B, where the gripper is opened, repeatedly, until the number of repetitions is exhausted. The task could be described as a grab and drop move.
- Perform sequential task: The robot arms perform the same task as above, but each will complete its move prior to the next robot in sequence begins its move. This routine tests the timeliness of message passing and the proper execution order of movement logic.
- Open (Close) gripper: Opens (closes) all end effectors.
- Check system status: The user is presented with submenus that gather the particular robot to address and the status types: system status, error stack, motor status, system configuration, diagnostics display.

V. SIMULATOR AND SIMULATION RESULTS

The µVision4 IDE is a window-based software development platform that combines a robust and modern editor, project manager, and makes facility. µVision4 integrates all the tools you need to develop embedded applications including C/C++ compiler, macro assembler, linker/locator, and a HEX file generator. µVision4 helps expedite the development process of your embedded application by providing the following:

- Full-featured source code editor,
- Device Database for configuring the development tool
- Project Manager for creating and maintaining your projects
- Integrated Make Utility functionality for assembling, compiling, and linking your embedded applications
- Dialogs for all development environment settings
- True integrated source-level and assembler-level Debugger with high-speed CPU and peripheral Simulator,
- Advanced GDI interface for software debugging in the target hardware and for connecting to the Keil ULINK Adapter family
- Flash programming utility for downloading the application program into Flash ROM
- Links to manuals, on-line help, device datasheets, and user guides.

The µVision4 IDE offers numerous features and advantages that help you to develop embedded applications quickly and successfully. The Keil tools are easy to use, and are guaranteed to help you achieve your design goals in a timely manner.

The CRACAN system works with the following functions. End users can select the options on the main menu as a guide, where the menu options are shown: Hard-home all robots, Move to point A, Move to point B, Perform coordinated task, Perform sequential task, Open gripper, Close gripper.

• Hard-home all robots. In this case the main controller will send data “0” to all robots.

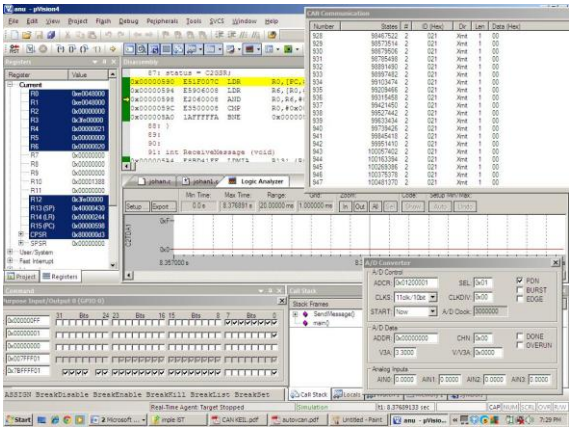


Fig 2: Block diagram of system

• Move to point A: This moves a user-named robot to position "A" for that the main controller will send data “A” Over CAN bus.

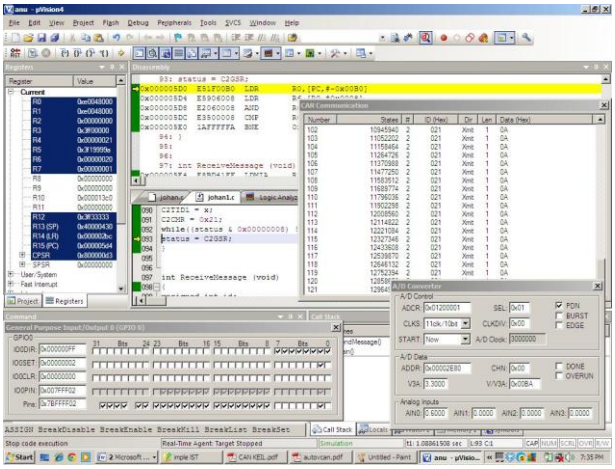


Fig 3: Block diagram of system

• Move to point B: This moves a user-named robot to position "B" for that the main controller will send data “B” Over CAN bus.

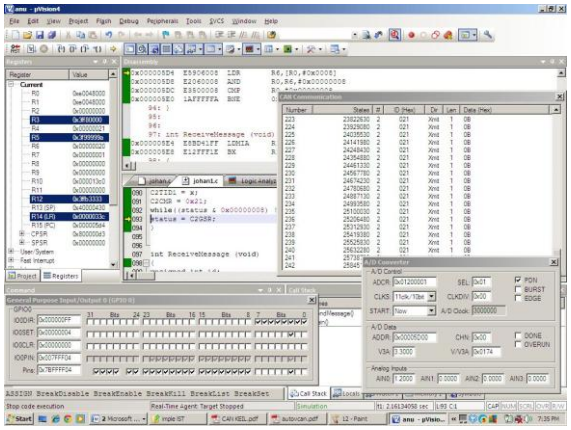
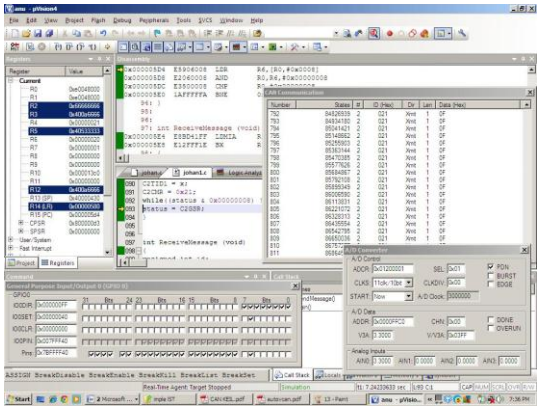


Fig 4: Block diagram of system

- Perform coordinated task: This option simulates a coordinated job, representative of an assembly line. All robots perform simultaneously. The user supplies the number of repetitions of the "job" to be performed, which consists of moving to point A, where the gripper is closed, to point B, where the gripper is opened, repeatedly, until the number of repetitions is exhausted. The task could be described as a grab and drop move.

- Perform sequential task: The robot arms perform the same task as above, but each will complete its move prior to the next robot in sequence begins its move. This routine tests the timeliness of message passing and the proper execution order of movement logic.

- Open gripper: Opens all end effectors. The main controller will send data “F” over CAN bus



. Fig 5: Block diagram of system

VI. CONCLUSION

Distributed control systems over CAN is designed and implemented so as to understand more and to find a better way to control various components of the real-time systems efficiently. On the distributed control systems, CRACAN, components are handled with event triggered approach to exchange messages and function properly. The discussed system is implemented over the controller area network (CAN) to have several robots function in synchronous and asynchronous ways. In synchronized move the robots of the CRACAN perform the same activities via control of the main microcontroller, which handles serial port initiation, robot initiation and others.

REFERENCES

[1] Haklin Kimm “Distributed Event-Triggered Robot Control System over Controller Area Network”, Proceedings the 2012 IEEE International

Conference on Industrial Technology, Athens, Greece, March 19 – 21, 2012.

[2] V. Claesson, C. Ekelin, N. Suri, "The event-triggered and time-triggered medium-access methods," Proceedings of the IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03), 2003.

[3] M. Bago, S. Marijan, and N. Peric, "Modeling Controller Area Network Communication", The 5th IEEE International Conference on Industrial Informatics, Vol. 1,pp.485-490, 23-27 June 2007.

[4] Magnenat S., "A Modular Architecture for Event-based Controls of Complex Robots," International Conference on Intelligent Robots and Systems," IEEE/ASME Transactions on Mechatronics, Vol. 16-2, April 2011.

[5] CAN specification version 2.0. Robert Bosch GmbH, Stuttgart, Germany,1991.

◈◈◈